UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/616,809 | 07/14/2000 | James Richard Wason | 13679(END9-2000-0080US1) | 6597 |

7590 09/28/2006

Richard L Catania Esq
Scully Scott Murphy & Presser
400 Garden City Plaza
Garden City, NY 11530

| EXAMINER | |
|---|---|
| CAMPBELL, JOSHUA D | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2178 | |

DATE MAILED: 09/28/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov



# BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/616,809
Filing Date: July 14, 2000
Appellant(s): WASON, JAMES RICHARD

John S. Sensny
For Appellant

## EXAMINER'S ANSWER

This is in response to the appeal brief filed July 5, 2006, appealing from the Office action mailed December 23, 2005.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-3 and 4-17 are rejected under 35 U.S.C. 102(e) as being anticipated by Andrews et al. (hereinafter Andrews, US Patent Number 6,317,871, filed on July 17, 1998).

**Regarding independent claim 1,** Andrews discloses a method in which a plurality of templates having literal fragments of a text file is generated (column 7, line 65-column 9, line 50 of Andrews, examiner points to column 4, lines 1-60 for the definitions of the terms used in Andrews). A macro class exists in which maps data from the text file to the computer application (column 7, line 65-column 9, line 50 of Andrews). A pointer to the macro class is embedded in one of the templates, that template being used as a prototype to generate a segment of an output file (column 7, line 65-column 9, line 50 of Andrews). When the pointer is reached in one of the templates, the pointer invokes the macro class and it uses the class to map data from one of the segments to the application and the macro invokes another template to further process the text file (column 7, line 65-column 9, line 50 of Andrews).

**Regarding dependent claim 2,** Andrews discloses a method in which the macros class reads a segment of the text file and uses the segment to initiate application update processing (column 7, line 65-column 9, line 50 of Andrews).

**Regarding dependent claim 3,** Andrews discloses a method in which the macros class derives data from the application and formats it into the text file (column 7, line 65-column 9, line 50 of Andrews).

**Regarding dependent claim 5,** Andrews discloses a method in which a controller is used that prevents structure clashes by placing text and data into appropriate places in a complex object structure as the text file is processed (column 7, line 65-column 9, line 50 of Andrews)

**Regarding dependent claim 14,** Andrews discloses a method in which the pointer that invokes the macro class also passes another template name, that template name is used invoke another template to process the text file (column 7, line 65-column 9, line 50 of Andrews).

**Regarding independent claim 6 and dependent claims 7-9 and 15,** the claims incorporate substantially similar subject matter as claims 1-3, 5, and 14. Thus, the claims are rejected along the same rationale as claims 1-3, 5, and 14.

**Regarding independent claim 10 and dependent claims 11-13 and 16,** the claims incorporate substantially similar subject matter as claims 1-3, 5, and 14. Thus, the claims are rejected along the same rationale as claims 1-3, 5, and 14.

**Regarding dependent claim 17,** Andrews discloses that the controller sets up a complex object structure and places the text data into that structure, and when the entire text file is processed that structure is used to process updating data into the application (column 7, line 65-column 9, line 50 of Andrews).

**(10) Response to Argument**

Regarding the appellant's arguments in reference to the independent claims, found on pages 12 and 13 of the appeal brief, the examiner maintains that the rejection is proper. The appellant is specifically arguing that Andrews does not disclose using a macro, which was invoked by one template, to map data from a text file to a computer application and to invoke another template to further process the text file. For the purpose of simplifying the response the argument will be broken down into two distinct parts, much like in the claim language. The first part of the argument is whether or not Andrews discloses that a macro, which was invoked by one template, maps data from a text file to a computer application. The second part of the argument is whether or not Andrews discloses that a macro, which was invoked by one template, invokes another template to further process the text file.

The first part of the argument is based on the claim limitation, "...using said pointer to invoke said macro class and using said macro class to map data from one of the segments of the text file to the computer application." A macro is defined by Andrews to be a complicated function of several lines of text that is invoked whenever a designated symbol is used anywhere within the computer program (column 1, lines 58-66 of Andrews). Andrews shows an example in which the symbol "ID" is a macro for a complicated algorithm, thus wherever the symbol "ID" appears in the program it will be substituted for its definition. This process of substitution is known as macro expansion (column 1, lines 63-66 of Andrews), which is based on the idea that when the macro is invoked using the during processing it is expanded to insert more complicated lines of

text that the symbol represents. The definition provided by Andrews directly corresponds to the accepted meaning of the word macro in the art at the time the invention was made. For instance, the Free On-Line Dictionary of Computing (hereinafter FOLDOC) provides the definition of macro to be, "A name (possibly followed by a formal argument list) that is equated to a text or symbolic expression to which it is to be expanded (possibly with the substitution of actual arguments) by a macro expander," (see Appendix A, provided by examiner). Thus, the definition of the word macro alone provides a basis that a macro maps segments of data to a computer application, as stated in the claim. Andrews does not teach away from this definition at any point, rather Andrews discloses the use of macros for mapping text segments from the text file to the translator application.

Andrews explains that when the invocation syntax of a macro is recognized, the text which the invocation syntax represents is substituted at that point, again this invocation of the macro is known as macro expansion (column 4, 32-41 of Andrews). In a more specific embodiment Andrews discloses that a "fragment tree" is used as a structure for defining partition templates (column 8, lines 10-51 of Andrews). The fragment tree is made up of cascading partition templates, the partition templates are represented by the word "part'n" in Figures 5 and 6. As can be seen in Figure 6, the fragment tree can be looked at as four initial partition templates. The fourth partition template of the initial four templates contains a macro with invocation syntax (i.e. pointer) of "stuff()" (Figure 6 and column 8, lines 10-51 of Andrews). This macro must be invoked and expanded in order to continue in the text processing. The expansion of

the macro requires the contents of the macro body (which is "a+13" in the case of the "stuff()" macro) be substituted for the macro invocation text (column 8, lines 10-51 of Andrews), in this case the text segment "a+13" was mapped to the computer application by the macro invocation text "stuff()". Thus, the rejection is proper because Andrews clearly discloses mapping data from one of the segments of the text file to the computer application.

The second part of the argument is based on the claim limitation, "...said macro class then invoking another on of the template to further process the text file." As clearly shown in Figure 6 of Andrews, the expansion process requires the macro to invoke another partition template in order to properly expand all of the parameter fragments (labeled "param fragments" in Figure 6) during the macro expansion process (Figure 6 and column 8, lines 10-51 of Andrews). The macro maps the data which is to be expanded, but in order to further process the text file the partition template ("part'n") for the parameter fragment ("param fragment") must be invoked (Figure 6 and column 8, lines 10-51 of Andrews). Thus, the rejection is proper because Andrews clearly discloses the macro invoking a template during the macro expansion process in order to properly substitute the mapped data of the macro body.

Regarding the appellant's arguments in reference to claims 14-16, found on pages 13 and 14 of the appeal brief, the examiner maintains that the rejection is proper. The appellant has not provided any additional arguments for these claims, rather the appellant has simply stated, based on the belief that the limitations regarding template invocation of the independent claims are not properly rejected, that the more specific

limitations regarding the same process as the limitations of the independent claims are also improperly rejected. However, the examiner has shown in the response to arguments for the independent claims that the rejection remains proper, thus the appellant's arguments for claims 14-16 amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. Thus, the rejection of these claims remains proper.

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Joshua D. Campbell

STEPHEN HONG
SUPERVISORY PATENT EXAMINER

Conferees:

Stephen Hong, Supervisory Patent Examiner for Group Art Unit 2178

Heather Herndon, Supervisory Patent Examiner for Group Art Unit 2176

# macro

A name (possibly followed by a formal argument list) that is equated to a text or symbolic expression to which it is to be expanded (possibly with the substitution of actual arguments) by a macro expander.

The term "macro" originated in early assemblers, which encouraged the use of macros as a structuring and information-hiding device. During the early 1970s, macro assemblers became ubiquitous, and sometimes quite as powerful and expensive as HLLs, only to fall from favour as improving compiler technology marginalised assembly language programming (see languages of choice). Nowadays the term is most often used in connection with the C preprocessor, Lisp, or one of several special-purpose languages built around a macro-expansion facility (such as TeX or Unix's troff suite).

Indeed, the meaning has drifted enough that the collective "macros" is now sometimes used for code in any special-purpose application control language (whether or not the language is actually translated by text expansion), and for macro-like entities such as the "keyboard macros" supported in some text editors (and PC TSRs or Macintosh INIT/CDEV keyboard enhancers).

(1994-12-06)

Try this search on Wikipedia, OneLook, Google

**Nearby terms:** Mac Playmate « MacPPP « MACRO « **macro** » macro- » macrology » Macromedia

Appendix A